Design Viewpoint

A Holistic Approach to Software Outsourcing

Steve Maylish • *Chief Commercial Officer, Fusion Biotec*
Frances Cohen • *President, Promenade Software Inc.*

The days of basic, minimally functional devices being accepted in the medtech market are long gone. Like it or not, today's users expect their medical devices to be like their iPhones or tablet computers. Screen buttons should animate when pressed, lists should scroll gracefully, and graphics should be attractive. Also, usability engineering is no longer an option. Human-computer interaction must now be considered, incorporated into the product design, and blessed by the U.S, Food and Drug Administration (FDA). A well-designed user interface/user experience (UI/UX) is essential for increasing safety and reducing errors.

As the Internet of Things (IoT) continues to grow, so too is software expectations; users want remote accessibility, easy upgradability, and solid security in their programs. Software projects now need more expertise than ever before, especially on complex devices. It's not uncommon to have embedded controls, applications, and test engineers working closely with graphical user interface (GUI) designers, web developers, and security experts. Software development is certainly a tall order to fill these days, particularly for ordinary companies with limited resources. Hence, it is best to outsource either some or all software development.

Outsourced software providers are important to a company's success. But for many firms, software engineering can be opaque, complicated, and confusing. That is why it's necessary to recognize the key capabilities of a good medical device software provider.

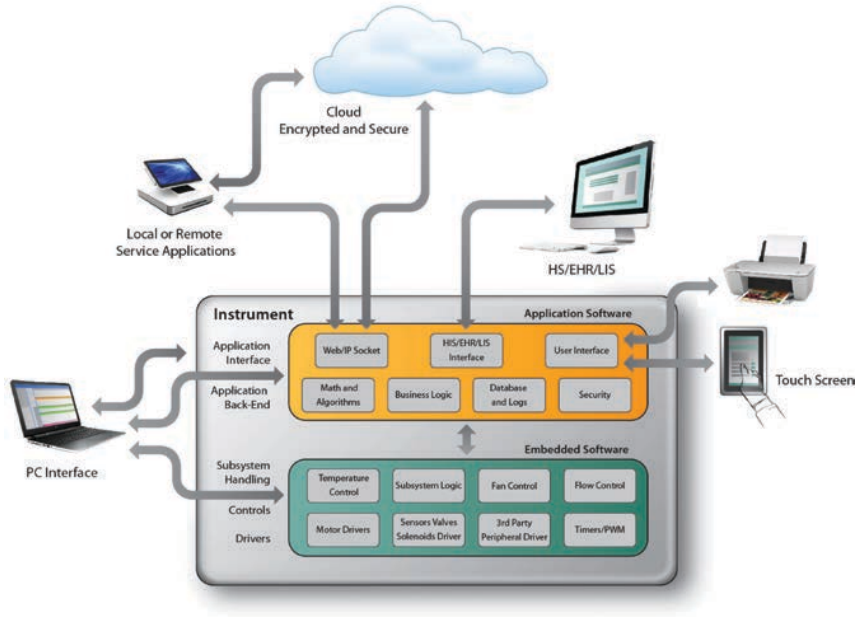
For starters, software should not be considered a static item eventually loaded into a medical device to make it work well. Instead, software should be viewed dynamically or holistically.

Creating software for complex instrumentation requires considerable planning. First, there are software requirements for the instrument. There are also requirements for the product's healthcare environment (hospital information system, electronic health record, lab information system, etc.) and the related connectivity (Bluetooth, Wifi, LAN). Lastly, there are software requirements for a device's sustaining environment (manufacturing, monitoring, updating, and service). No other engineering discipline is so embedded in a business.

Product software can be divided into two general categories—embedded and application operating systems. Embedded software is hidden from the user, and features fixed hardware requirements and capabilities such as controlling sensors, valves, or motors. Embedded software includes device drivers, which are written for specific hardware. It's also highly dependent on the microprocessor chosen. Because of this, embedded software engineers need a comprehensive knowledge of electronics as well as an understanding of real-time firmware engineering.

Application software, on the other hand, typically runs for the user's benefit. It runs on top of the embedded software and is just as important. Most users are familiar with application software and understand the value of an intuitive user interface. Application software defines the instrument for the user, allows for communication to the outside world, and incorporates unique technology. Application software is more likely to be revised and updated over time.

Choosing a provider with the appropriate engineering skill is important, because developing the algorithms, GUI, database, or real-time embedded control takes different skills. Companies should understand the necessary product features before choosing a software provider. Generally, IT specialists are not familiar with embedded software. Likewise, embedded software and firmware engineers typically don't work on PC applications. There is no way to determine cost without a clear vision and understanding of the medical



device's features. Changing those features or the overall direction of a product during development can significantly impact schedules and budgets. And negative cash flows can trigger disputes.

Any experienced provider should not be starting software development from scratch. The more disciplined they are, the more well-defined their codebase frameworks should be, both for embedded and application software. Projects are unique in their technology applications, but not in their framework. The framework is more mundane, but it's the infrastructure providing the baseline for quality instrument software. Outsourced providers should have an established framework and baseline design available, tested, and proven.

Companies should also ask software providers to explain their solutions, particularly their software design methodology. Reworking old code to fit a particular project is not good software engineering. A provider should enthusiastically discuss software design, frameworks, and the merits of his or her design process. Ask lots of questions. The devil is in the details.

Since software engineering is another key to success, providers should have top-notch engineers with applicable university degrees and/or years of experience in software and medical device development. A desirable team should have senior and junior engineers, providing a mix of new technologies, ideas, and energy with experienced oversight. The team focus should be on robust quality and safety, because poor design will eventually result in unmaintainable and buggy software.

Because software is not static, emphasizing maintainability of the code allows other engineers to later alter it without introducing bugs. Design and code reviews should be the norm. This ensures the code is clean, modular, and maintainable for any future team. Deliverables should include full and complete source code, along with build instructions, details about the tools used, and any license agreements. Companies may later build an in-house team, or chose a different software provider. This gives medtech firms options to maintain and support their product in the future, and avoid provider exclusivity.

Testing is critical for robust software. It's safe to assume that untested software is broken. Even the best programmers introduce bugs into their code. Providers that don't test with rigor and discipline throughout the software development lifecycle risk impacting product quality. Testing should incorporate more than final verification and validation; it should include unit and integration testing while code is being developed.

Rick D. Craig and Stefan P. Jaskiel state in their book, "Systematic Software Testing" that "Testing is a concurrent lifecycle process of engineering, using and maintaining testware in order to measure and improve the quality of software being tested." Companies should ask their software providers for their test strategy. What tools have they developed to make testing easier? Can regression testing be automated? Can faults be injected, and behavior checked? By the end of development, a robust suite of automated tests should verify that changes do not introduce errors. These considerations should all be part of a provider's development package.

Generally speaking, a Gantt chart project plan is valuable for tracking milestones (waterfall process), but isn't always practical from a weekly perspective. Development is often a discovery process, requirements change throughout a project, and software is sometimes required to compensate for hardware limitations later in the project. Because of these realities, software providers should have an iterative development methodology. Despite the dynamic nature of the iterative process, the FDA has embraced agile Scrum as a lifecycle model, having accepted AAMI TIR 45 (Use of Agile Methods).

Scrum accepts the reality of software development and creates a methodology where change is expected and handled. At the same time, it provides visibility for stakeholders into the progress. Agile Scrum breaks development into small, manageable portions or two- to three-week "sprints." Here, the work is prioritized, defined for the interval, executed, and demonstrated to stakeholders upon completion. This type of process or any similar iterative one is much better suited to the reality of software development.

Agile Scrum provides stakeholders with a successful development path by incorporating a prioritization process. As features make their way into the requirements specification, the product owner (representing the stakeholders) prioritizes them. Many of these features can wait for a follow-on upgrade and are not worth delaying the product launch. Since software features are developed in order of their priority, this allows for a minimal viable product in the least amount of time. At any point, companies can decide whether it is worth delaying the product launch for

10 Questions to Ask a Potential Software Provider

-1. What is your experience in embedded control software?
2. What is your experience with application software?
3. Do you have experience implementing graphical user interfaces?
4. How do you ensure that the code base is maintainable/reusable?
5. What is your software test strategy?
6. Can you simulate hardware with your software?
7. What tools do you provide for experimentation/validation?
8. How do you manage cyber security and data encryption?
9. How familiar are you with regulatory requirements?
10. Can your software support manufacturing and field service testing?

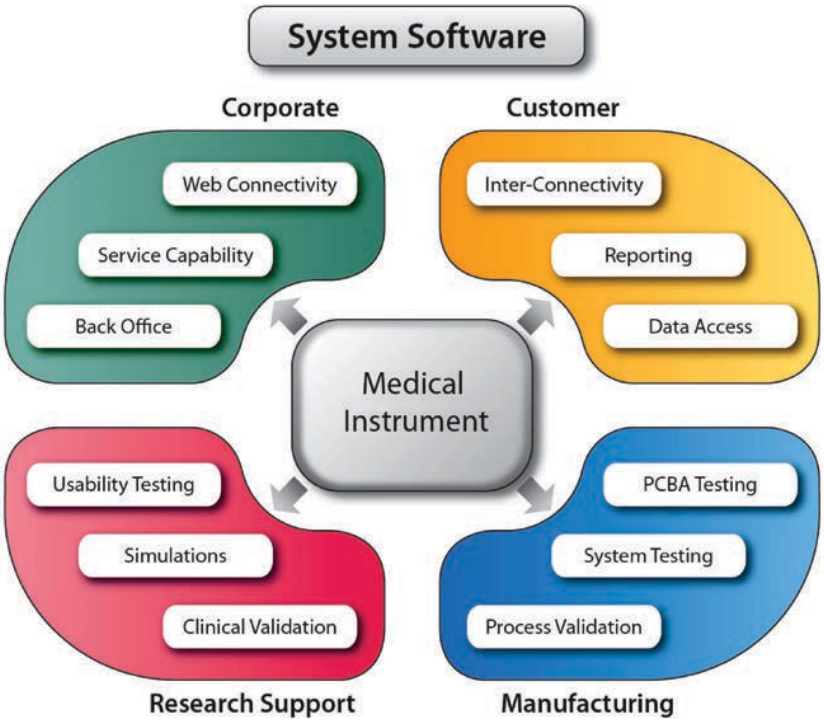
the remaining nice-to-have features.

Once instrument software requirements are understood, companies should define the instrument environment: corporate, customer, research, and manufacturing. The corporate and customer software environments should be defined in a marketing requirements document for the instrument. This is typically driven by marketing efforts and user focus groups.

On the corporate side, companies should determine how much connectivity is needed with a particular device and customers. Collecting data on product use can be strategically very valuable. Tracking how the device is used allows companies to better understand customers' needs, how to service the instrument, and the kinds of new technologies to improve performance. Companies can provide more value to customers by updating their software, thereby providing additional marketing and sales opportunities. Customers can be charged based on usage to provide a recurring revenue stream, keep track of consumable needs for automated replenishment, or provide cloud-based data storage and information on devices used in the field. If those features are not immediate concerns, then software providers must be able to plan for those needs in the future.

On the customer side, medtech firms should consider networking with the hospital information system or electronic health record, interconnecting multiple instruments, other medical instruments, or consumer devices that give more value to customers. Some may even consider the home healthcare market or wearables. Regardless of market entry, device software can differentiate a company from the competition, provide access to a larger customer base, and deliver data-based analytics.

For complex instruments, research support can be vital. This support can come early with usability testing, system simulation, and GUI review, or later with final clinical validation. Usability testing should be performed with the GUI before the product reaches the prototype stage. The hardware may need to be simulated to optimize system protocols. Simulation allows software development to work in parallel with other disciplines, saving development time. System engineers and scientists can modify parameters or chemistries during development in order to respond to new mechanical systems, for example.



Issues between software, mechanical, electrical, chemistry, optics, and physics must be worked out—ideally, by software that's designed for experimentation and testing. Without such a program, a dedicated software engineer might be needed just to support system integration. A provider experienced in software design can help companies avoid such a middleman by creating a reasonably easy method to allow non-software staff to change parameters and experiment. Companies should ask about the available tools for experimentation and research support.

Once the product reaches manufacturing, software can also play an important role. Software can be used to validate the manufacturing process and it can also be used to test sub-systems, test electronic circuit boards, burn-in the instrument, or troubleshoot the device. Software providers should be able to support these activities with either scripts or custom applications, ideally without undue costs. With a good application programming interface or set of software routines, this capability can be built in while the software code is written. As such, test routines are easy to incorporate later.

On the regulatory side, medical device software is governed by FDA's software guidance and IEC 62304. Outsourced providers must prepare software documentation per the regulatory requirements. They should create software requirements with their partner's input, including functional, interface, performance, data storage, safety, and security mandates. The requirements must be correct, unambiguous, complete, consistent, verifiable, and traceable. Writing useful, detailed requirements is a discipline that necessitates software expertise.

Software risk management is also a very deliberate process. It has to trace potential hazards to the components of the software system, and to risk control measures implemented in software. Software

of Unknown Provenance (SOUP) (third-party software-like operating systems) needs a detailed risk evaluation. Other ingredients include a software development plan, configuration management plan, software architecture, software design, verification activities, and formal test plans, protocols, and reports. In short, there must be a full suite of documents based upon software regulatory expertise, and the software provider should provide them as part of its services.

Medical devices are now very software-centric. This affects development, research, manufacturing, and commercialization. Software has exploded with new technologies in the last decade, and the right software provider can exploit these to its partner's advantage. Software not only runs an instrument, it also runs corporate business strategies and profits. As the Internet of Things continues to grow, so will medical device software. Modern software, consequently requires a different mindset. Remember, software is a dynamic part of any medical product and a company's success. ❖

Steve Maylish has been part of the medical device community for more than 30 years. He is currently chief commercial officer for Fusion Biotec, an Irvine, Calif.-based contract engineering firm that brings together art, science, and engineering to create medical devices. Early in his career, Maylish held positions at Fortune 100 corporations such as Johnson & Johnson, Shiley, Sorin Group, Baxter Healthcare, and Edwards Lifesciences. After receiving his MBA in 2002, Maylish focused his business development skills on two boutique medical design/manufacturing firms that were acquired. Afterwards, he founded Maytrix Medical to provide business development to third-party engineering firms. Maylish has an A.S. degree in electronics from RETS Electronics Institute, a B.S. in business from the University of La Verne, and an MBA from the University of California, Irvine.

Frances Cohen has more than 15 years of experience leading software teams for medical device software. Starting with heart defibrillators for Cardiac Science and GE Healthcare and following with Source Scientific LLC and BIT Analytical Instruments Inc., Cohen has overseen dozens of projects through development and the FDA, including IDEs, 510(k)s, and pre-market applications. Cohen is president of Promenade Software Inc., a software services firm specializing in medical device critical safety systems. The company provides device software for various instruments, from wearables and mobile devices to point-of-care and complex in-vitro diagnostic systems. She has a B.S. in computer engineering from the Technion in Israel.